修士学位論文

題目

時間抽象を行う洗練手法を用いた
確率時間システムの到達可能性解析手法

指導教員
楠本 真二 教授

報告者
伊藤 明彦

平成 22 年 2 月 8 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻

平成 21 年度 修士学位論文

時間抽象を行う洗練手法を用いた確率時間システムの到達可能性解析手法

伊藤 明彦

内容梗概

モデル検査は，システムの設計段階での不具合の発見に有用である．そのため，システムの高信頼設計において適用されるようになってきている．ネットワークシステムのような，振る舞いに確率的な要素を含むシステムの場合，そのシステムが不具合を示す確率を計測することが非常に重要であり，そのシステムの信頼性を計測するために必要不可欠な性質であると考えられる．さらにそれが実時間のシステムであれば，時間的な制約を持ち，状態数が時間制約に応じて指数的に増加するため，検査に対するスケーラビリティに大きな制約が加わる．一方，Clarke らが古典的な Kripke 構造を持つオートマトンに対して反例に基づく抽象化洗練 CEGAR を提案し，状態爆発を回避しながらモデル検査を自動で行う枠組みを提案した．この枠組みは時間オートマトンをはじめとして，多くのオートマトンモデルに対しての有用性を示した．

既存研究として，時間オートマトンに対する抽象化洗練手法が提案されている．その手法に基づいて，本研究では確率時間オートマトンの CEGAR を用いた到達可能性解析を行う手法を提案する．ただし，確率時間システムでは，反例となるパスの組み合わせが各ロケーションにおける時間経過によって変動するため，単純に複数個のパスの探索では正しい反例を提示できない．本研究では，そのパスの組み合わせを決定するために，元モデルの等価変換を行うことで解決している．

本論文では，実際の確率時間オートマトンに対して適用し，提案手法の概要を示したのち，また提案手法の具体的なアルゴリズムを形式的に記述し，提案手法の正当性を示す．さらに，提案手法の部分的な実装を行い，状態数や実行時間などで，手法について比較考察を行う．最後に，今後の課題とまとめについて述べる．

主な用語

確率時間オートマトン, 時間オートマトン, マルコフ決定過程, 離散時間マルコフ連鎖, CEGAR ループ, モデル検査，到達可能性解析

目 次

図 目 次

表 目 次

1　まえがき

近年, 情報システムの高信頼設計にモデル検査 [4][8] を利用することが望まれている．モデル検査によって，設計されたシステムのシステムの振る舞いに関する不具合を早期に発見できることが期待されるためである．モデル検査は，システムを有限の状態遷移系として形式的に記述し，また検査を実行する性質を論理式で入力することで，状態遷移系の全状態を網羅的に探索する．それにより，設計されたシステムが仕様（性質）を満たしているかどうかを保証する手法である. しかし, 大規模なシステムに対しては状態数爆発を起こすなど, スケーラビリティの低さが課題となっている．モデル検査のスケーラビリティの低さを改善する手法として, 検査する性質ごとにモデルの状態数を適切に削減し，さらにはその抽象化の度合いを自動的にコントロールするモデル抽象化手法が注目されている [17][29]．とりわけ，モデル検査時における反例を用いて抽象度を自動コントロールする CEGAR(Counter Example Guided Abstraction Refinement) ループが大きな注目を浴びている [7].

実際に検査対象となるモデルとしては，ネットワークプロトコルなどのランダム性を持つモデルを検証する場合，確率モデルが用いられる．確率モデルとは，有限の状態遷移系に, 状態遷移確率を付加したモデルである．確率モデルの最も代表的なものに，マルコフ決定過程 [28] が挙げられる．確率モデルを検査するための代表的なツールとして，確率的モデル検査ツール PRISM[16] があり，これを用いて実際のネットワークシステムの検証が広く行われている [12][21][24].

さらに，この確率モデルに実時間の性質が加わった確率時間システムの動作検証には，確率時間オートマトンと呼ばれるモデルが用いられる．確率時間オートマトンでは, 有限のロケーションと呼ばれる状態に，実数値をとるクロック変数を用いた制約が付加されるため, 確率時間オートマトンは無限の状態空間を持つことになる．モデル抽象化を行わない従来のモデル検査では時間領域が実質有限個に抑えられることを利用し有限状態のモデルに対し検査を行う．しかし，この状態数はロケーションやクロックの個数に対して指数的に増加するため，状態数を適切に縮小するモデル抽象化手法が必要となる．本研究では，文献 [22] で定義されている一般的な確率時間オートマトンを対象としており，時間オートマトンにおけるクロック変数を除去する時間抽象を行う抽象化洗練手法 [23][25][26] を，確率時間オートマトンの抽象化洗練手法として拡張した手法を提案する．確率時間オートマトンの時間抽象化を行うことで，確率モデルに変換し，確率モデルの反例を探索する手法 [1][13] に基づいて反例を探索することができる.

ただし，確率時間オートマトンの一般的な性質に対する反例というのは，時間オートマトンにおける CEGAR とは異なり，確率的な性質を検査したい場合，性質を満たさないパス 1 つだけでなく，反例が複数個のパスを含む場合が存在する．このため，パスが複数個にわた

る場合，反例となるパス群の同時実行可能性について考慮しなければならない．確率時間オートマトンでは非決定的に時間経過が行われるため，それらのパスが同時に実行されることは無いからである．

この問題があるため，単純に従来の CEGAR の手法を適用できない．よって，提案手法の中で，元の確率時間オートマトンの変換を繰り返し，同時実行不可能なパスを明確化することで，抽象モデル上で同時に発見されないようにする変換手法を考案した．

よって，本手法を利用することで，CEGAR の枠組みを用いた確率時間オートマトンのモデル検査を実行することができる．本研究では，手法のために必要なアルゴリズムの定義を行い，手法の理論的な正しさに関する議論を述べている．さらに，提案手法を実現するツールのプロトタイプを作成し，評価を行った．実験として，既存手法の一つである Digital cloks[19] との比較を行った．実験対象として用いたモデルは，FireWire Root Contention Protocol[15], である．実験結果として，状態数を大幅に削減し，実行時間も削減することができた．

以下 2．では，まずモデルとして利用されるオートマトン（確率オートマトン，時間オートマトン，確率時間オートマトン）について述べる．また，本研究でも利用する CEGAR ループについて，さらにモデル検査によって調べられる性質について述べる．次の 4．では，確率時間オートマトンに対する一般的なモデル検査についてと，その検査を行う既存手法，さらに本研究で提案する手法について説明する．5．では，提案手法を実装し，実際のモデルへの適用実験を行った結果を紹介する．6．では，5．で行った結果を元に，本研究での提案手法の有用性について考察する．

2 準備

2.1 確率分布

有限集合 Q 上の（離散的な）確率分布は関数 $\mu : Q \to [0,1]$ によって割り当てられ，$\sum_{q \in Q} \mu(q) = 1$ である．$\mathsf{support}(\mu)$ を $q \in \mathsf{support}(\mu)$ である Q の部分集合とし，かつその時に限り $\mu(q) > 0$ である．無限集合 Q_∞ において，$\mathsf{Dist}(Q_\infty)$ を Q_∞ の有限部分集合上の確率分布の集合とする．

2.2 確率オートマトン

確率オートマトン [27] は，有限状態数のオートマトンに対し，状態遷移を行う条件に，確率の値を付加しているものである．確率オートマトンは，以下 3 つのオートマトンに大別される．

- 離散時間マルコフ連鎖 (DTMC:Discrete Timed Markov Chain)
- 連続時間マルコフ連鎖 (CTMC:Continuous Timed Markov Chain)
- マルコフ決定過程 (MDP:Markov Desicion Process)

2.2.1 離散時間マルコフ連鎖 (DTMC:Discrete Timed Markov Chain)

定義 2.1 (離散時間マルコフ連鎖の構文)**.** マルコフ連鎖 DTMC は *4* 項組 $\mathsf{DTMC} = (S, \bar{s}, \mathbf{P}, L)$ によって定義される．

- S：状態の有限集合
- $\bar{s} \in S$：初期状態
- $\mathbf{P} : S \times S \to [0,1]$：確率遷移行列
- $L : S \to 2^{AP}$：ラベル付け関数

要素 $\mathbf{P}$ は，状態 s から $\prime s$ への遷移確率を与える行列である．$s \in S$ であるすべての状態に対し，$\sum_{s \in S} \mathbf{P} = 1$ である必要がある．終端となる状態は，自身へのループ（確率が 1 で同じ状態への遷移）を付加することでモデル化される．ラベル付け関数 L は，AP 中の原子命題式を状態から状態への写像を与える．

2.2.2 マルコフ決定過程 (MDP:Markov Desicion Process)

定義 2.2 (マルコフ決定過程の構文)**.** マルコフ決定過程 MDP は以下の *4* 項組 $\mathsf{MDP} = (S, \bar{s}, Steps, L)$ によって定義される.

- S：状態の有限集合
- $\bar{s} \in S$：初期状態
- $Steps : S \times S \to [0, 1]$：遷移関数
- $L : S \to 2^{AP}$：ラベル付け関数

集合 S，初期状態 $\bar{s}$ とラベル付け関数 L は，DTMC と同様である．確率遷移行列 $\mathbf{P}$ は，$Steps$ によって代替される．これは，各 $s \in S$ を $Dist(S)$ の空でない部分集合にマッピングする関数である．すなわち，すべての確率分布の集合である．直観的に，状態 $s \in S$ に対し，$Steps(S)$ の要素は,「非決定的な選択」を表している．各々の非決定的な選択は，他の状態への遷移の可能性を与える確率分布である.

2.2.3 連続時間マルコフ連鎖

定義 2.3 (連続時間マルコフ連鎖の構文)**.** 連続時間マルコフ連鎖 CTMC は以下の *4* 項組 $\mathsf{CTMC} = (S, \bar{s}, R, L)$ によって定義される.

- S：状態の有限集合
- $\bar{s} \in S$：初期状態
- $\mathbf{R} : S \times S \to$：遷移割合行列
- $L : S \to 2^{AP}$：ラベル付け関数

S,$\bar{s}$ と L に関しては DTMC と同様である．しかし，遷移割合行列 $\mathbf{R}$ に関しては，確率，でなく状態間の遷移を決める割合を与える.

2.3 確率オートマトンの性質

本節では，確率オートマトンの性質を明確化するための一般的な形式を示す．PCTL は DTMCs，MDPs によって解釈され，CSL は CTMCs によって解釈される.

2.3.1 PCTL

PCTL(Probabilistic Computational Tree Logic)[14] は，時相論理 CTL の確率的な拡張である．本来は，pCTL[3] と同様なものである．PCTL の文法は以下のように与えられる．

$$\phi \;:=\; true \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathcal{P}_{\bowtie p}[\psi] \tag{1}$$

$$\psi \;:=\; X\phi \mid \phi\,\mathcal{U}^{\leq k}\,\phi \mid \phi\,\mathcal{U}\,\phi \tag{2}$$

ここで，a は原子命題式，$\bowtie \in \{\leq, <, >, \geq\}, p \in [0,1]$ であり，$k \in \mathbb{N}$ である．PCTL 式は，DTMC, MDP 上で解釈される．原子命題式 a はこの DTMC, MDP の各状態にラベル付けするために使用される集合から用いられる．

これらの文法では，状態式 ϕ とパス式 ψ を区別しており，それぞれ状態，パス上において評価される．モデルに性質は，常に状態式において表現される．パス式は確率的なパスオペレータ $\mathcal{P}_{\bowtie p}[\psi]$ のパラメータとしてのみ表記される．直観的に，もし状態 s までのパスが ψ を満たしているならば，状態 s は $\mathcal{P}_{\bowtie p}[\psi]$ を満たす．

パス式として，$X(next)$，$\mathcal{U}^{\leq k}(until)$，$\mathcal{U}(bounded\,until)$ オペレータを用いている．これらは時相論理において基準となるものである．直観的に，ϕ が次の状態で満たされる場合に $X\phi$ が真となり，$\phi_1\,\mathcal{U}^{\leq k}\,\phi_2$ では，ϕ_2 が k ステップまでに満たされ，ϕ_1 がその状態まで真であり続ける場合に，真となる．$\phi_1\,\mathcal{U}^{\leq k}\,\phi_2$ は，ϕ_2 が任意の点で真となり，ϕ_1 がその状態まで真であり続ける場合に，真となる．

PCTL over DTMC

DTMC$(S, \bar{s}, \mathrm{P}, L)$，状態 $s \in S$ と PCTL 式 ϕ に対し，ϕ が s において満たされることを，$s \models \phi$ と表記する．また，$Sat(\phi)$ によって，ϕ を満たすすべての状態の集合 $\{s \in S \mid s \models \phi\}$ を示す．同様に，パス式 ψ を満たすパス ω に対し，$\omega \models \psi$ とする．ここで，DTMC に対する PCTL の意味定義を以下で与える．パス ω に対して，

$$\begin{aligned}
\omega \models X\phi \quad &\Leftrightarrow \quad \omega(1) \models \phi \\
\omega \models \phi_1\,\mathcal{U}^{\leq k}\,\phi_2 \quad &\Leftrightarrow \quad \exists i \leq k\,.\,(\omega(i) \models \phi_2 \wedge \omega(j) \models \phi_1\ \forall j < i) \\
\omega \models \phi_1\,\mathcal{U}\,\phi_2 \quad &\Leftrightarrow \quad \exists k \geq 0\,.\,\omega \models \phi_1\,\mathcal{U}^{\leq k}\,\phi_2
\end{aligned}$$

そして，状態 $s \in S$ に対し，

$$\begin{aligned}
s &\models true && \text{for all } s \in S \\
s &\models a && \Leftrightarrow \quad a \in L(s) \\
s &\models \phi_1 \wedge \phi_2 && \Leftrightarrow \quad s \models \phi_1 \wedge s \models \phi_2 \\
s &\models \neg\phi && \Leftrightarrow \quad s \not\models \phi \\
s &\models \mathcal{P}_{\bowtie p}[\psi] && \Leftrightarrow \quad p_s(\psi) \bowtie p
\end{aligned}$$

ここで，$p_s(\psi) = Prob(\{\omega \in Path_s \mid \omega \models \psi\})$ である．確率 $Prob_s$ は，パスの集合に確率を割り当てる．

PCTL over MDP

パスの集合の確率は，特定のアドバーサリに対し計算される．$p_s^A(\psi)$ によって，アドバーサリ A の元でパス式 ψ を満たす s からのパスの確率を示す．すなわち，$p_s^A(\psi) = Prob_s^A(\{\omega \in Path_s^A \mid \omega \models \psi\})$．セマンティクスを与えるために，アドバーサリ Adv のクラスを選択する．もしすべてのアドバーサリ $A \in Adv$ において $p_s^A(\psi) \bowtie p$ ならば，状態 s は $\mathcal{P}_{\bowtie p}[\psi]$ を満たす．それ故に，この充足関係はアドバーサリ Adv のクラスによってパラメータ化される．

$$\begin{aligned}
s &\models_{Adv} true && \text{for all } s \in S \\
s &\models_{Adv} a && \Leftrightarrow \quad a \in L(s) \\
s &\models_{Adv} \phi_1 \wedge \phi_2 && \Leftrightarrow \quad s \models \phi_1 \wedge s \models \phi_2 \\
s &\models_{Adv} \neg\phi && \Leftrightarrow \quad s \not\models \phi \\
s &\models_{Adv} \mathcal{P} && \Leftrightarrow \quad p_s^A(\psi) \bowtie p \; for\ all\ A \in Adv
\end{aligned}$$

2.3.2 CSL

CSL(Countinuous Sthochastic Logic) の論理は，PCTL と似ているが，CTMC モデルにおける性質を明確化するためのものである．

$$\phi \quad ::= \quad true \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathcal{P}_{\bowtie p}[\psi] \mid \mathcal{S}_{\bowtie p}[\phi] \tag{3}$$

$$\psi \quad ::= \quad X\phi \mid \phi\, \mathcal{U}^{\leq t}\, \phi \mid \phi\, \mathcal{U}\, \phi \tag{4}$$

ここで，a は原子命題式，$\bowtie \in \{\leq, <, >, \geq\}, p \in [0,1]$ であり，$t \in \mathbb{N}_{\leq 0}$ である．PCTL 式に対し，$\mathcal{P}_{\bowtie p}[\psi]$ は，制限 $\bowtie p$ を満たす状態から，パス式 ψ が満たされる確率を示す．パス式は，CSL に関しても PCTL と同様であるが，bounded until オペレータ $\phi_1 \mathcal{U}^{\leq t} \phi_2$ のパラメータ t においては，PCTL のように非負の整数ではなく，非負の実数である．この式は，ϕ_2 が

$[0,t]$ 内の任意の時間で満たされ，ϕ_1 が先行しているすべての瞬間において満たされることを意味する．混乱を避けるため，$time\text{-}boundeduntil$ としてこれを言及する．この S オペレータは，CTMC の定常状態の振る舞いを示す．式 $\mathcal{S}_{\bowtie p}[\phi]$ は，制限 $\bowtie p$ で，ϕ を満たす状態に到達する定常状態の確率を表明する．

2.4 確率的モデル検査

確率的モデル検査アルゴリズムは，PCTL over DTMCs，PCTL over MDPs，CSL over CTMCs の 3 つのタイプの検査が存在する．モデル検査アルゴリズムは，PCTL もしくは CSL の論理式 ϕ とその論理式に適合するモデルを用い，ϕ を満たす状態を含む集合 $SAT(\phi)$ を返す．

モデル検査アルゴリズムの大まかな構造は，3 つのケース共に同じであり，[6] によって示されている従来の CTL モデル検査アルゴリズムに由来する．最初に，式 ϕ の解析木の構成を行う．解析木における各ノードは，ϕ のサブ式によってラベル付けされる．そのルートノードは ϕ 自身によってラベル付けされ，木の葉っぱとなるノードは，$true$ もしくは原子命題 a によってラベル付けされる．木のルートに向かって，サブ式をそれぞれ満たす状態の集合を，再帰的に計算し，最終的に，各状態は ϕ を満たすかどうかを決定する．

2.5 マルコフ決定過程のモデル検査手法

マルコフ決定過程 MDP に対するのモデル検査手法のうち，代表的な手法として ValueIteration[5] が挙げられる．ValueIteration（Algorithm1 に示す）では，パラメータ g,f に設定された値に従って，到達可能性，安全性における最大確率値，最小確率値を計算する．例えば，$f = max$ であり，$g = min$ の場合に到達可能性の最大確率を計算する．さらに，各状態において，パラメータに適合するアクションが選択されるため，ValueIteration によって求められた確率を示すアドバーサリを求めることができる．

Algorithm 1 $\mathrm{ValIter}(T, f, g, \epsilon_{float})$

$v := [T]$
repeat
  $\hat{v} := v$
  for all $s \in S$ **do**
    $v(s) := f([T](s), g\{\sum_{s' \in S} p(s,a,s') \cdot \hat{v}(s') \mid a \in \Gamma(s)\})$
  end for
until $||v - \hat{v}|| \leq \epsilon_{float}$
return v

2.6 クロック変数とゾーン

確率時間オートマトンモデルでは，非負数の実数の集合 $\mathbb{R}$ 上の値である，クロック変数を使用する．クロック変数の有限集合 $\mathcal{X}$ を 1 つ定めるする．関数 $v : \mathcal{X} \to \mathbb{R}$ は，クロック評価として参照される．すべてのクロック評価の集合は，$\mathbb{R}^{\mathcal{X}}$ を意味する．また，すべてのクロック変数の値が 0 であるようなクロック評価を ν_0 とする．

任意の $v \in \mathbb{R}^{\mathcal{X}}$，$t \in \mathbb{R}$ と $X \subseteq \mathcal{X}$ に対し，t と，0 にリセットする X 中のクロックを評価するための $v[X := 0]$ および，すべてのクロック変数をインクリメントする v においてクロック評価を意味する，$v + t$ の 2 つを使用する．

$\mathcal{X}$ のゾーンの集合を $Zones(\mathcal{X})$ と記述し，文法的に以下のように定義する．

$$\zeta ::= x \sim c \mid x - y \sim c \mid \zeta \wedge \zeta \mid \mathtt{true}$$

ここで，$x, y \in \mathcal{X}$ であり，$\sim \in \{<, \leqslant, \geqslant\}$ である．ゾーン ζ は，ζ を満たす（$v \triangleright \zeta$ を意味する）クロック評価 v の集合を示す．すなわち，$v(x)$ で $\mathtt{true}$ となる各クロック x の代替として，ζ を用いる．

2.7 時間オートマトン

時間オートマトンは，有限状態数のオートマトンに，時間経過を表すクロック変数を付加したオートマトンである．本節では，時間オートマトンに関する定義を与えている

定義 2.4 (時間オートマトンの構文). 時間オートマトン TA は，次のタプル $\mathsf{TA} = (A, L, l_0, C, I, T)$ によって定義される．各項目は以下のように定義される．

- A：アクションの有限集合
- L：ロケーションの有限集合
- $l_0 \in L$：初期ロケーション
- C：クロック変数
- $I : L \to c(C)$：ロケーションからインバリアントへの写像
- T：遷移集合

遷移集合 T は $T \subset L \times A \times c(C) \times 2^C \times L$ を満たす.

遷移 $t = (l_1, a, g, r, l_2) \in T$ を $l_1 \xrightarrow{a,g,r} l_2$ と表記する．g をガード，r をリセットクロックと呼ぶ.

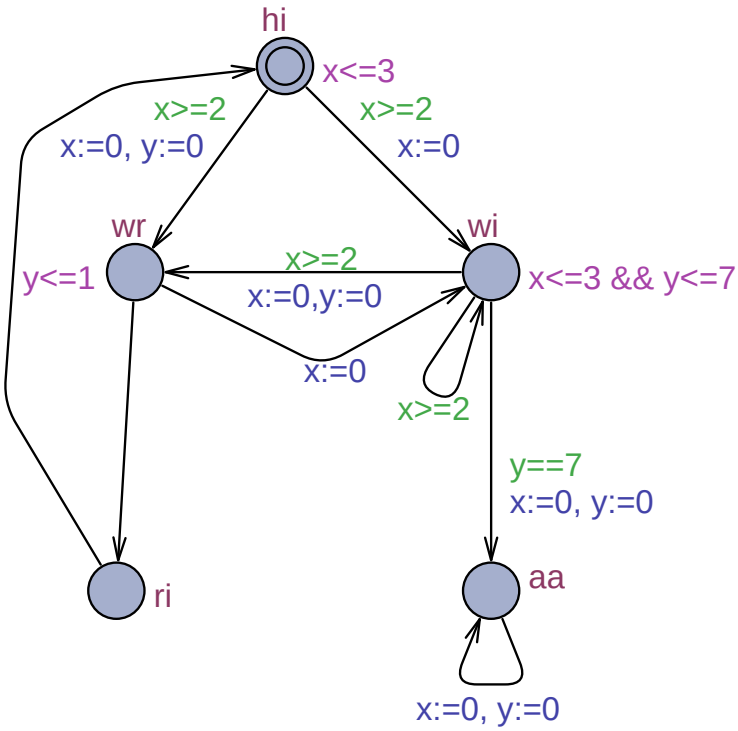

図 1: 時間オートマトン

定義 2.5 (クロックの評価関数)**.** $\nu : C \to \mathbb{R}_{\geq 0}$ なる ν をクロックの評価関数と呼ぶ.
$d \in \mathbb{R}_{\geq 0}$ に対して $(\nu + d)(x) = \nu(x) + d$ と定義する.
$r \in 2^C$ に対して $r(\nu) = \nu[x \mapsto 0], x \in r$ と定義する.
$g \in c(C)$ に対して $g(\nu)$ を ν の各クロックの評価のもとでの, g の評価と定義する.
また, 0^C で $x \in C$ なる各クロック x に対する値を 0 とするクロック評価関数を表すとする.
ν の全てからなる集合を $N \subseteq \mathbb{R}^C_{\geq 0}$ とする.

2.7.1 時間オートマトンの意味

定義 2.6 (時間オートマトンの意味)**.** 時間オートマトン $\mathscr{A} = (A, L, l_0, C, I, T)$ に対して
$\mathscr{A}$ の状態集合を $S = L \times N$ とする.
$\mathscr{A}$ の初期状態は $(l_0, 0^C) \in S$ で与えられる.
状態遷移 $l_1 \xrightarrow{a,g,r} l_2$ $(\in T)$ に対し，次の *2* つの遷移が定義される.

$$\frac{l_1 \xrightarrow{a,g,r} l_2, g(\nu), I(l_2)(r(\nu))}{(l_1, \nu) \overset{a}{\Rightarrow} (l_2, r(\nu))}$$

$$\frac{\forall d' \leq d \quad I(l_1)(\nu + d')}{(l_1, \nu) \overset{d}{\Rightarrow} (l_1, \nu + d)}$$

前者を離散遷移, 後者を時間遷移と呼ぶ.

定義 2.7 (時間オートマトンの意味モデル)**.** 時間オートマトン $\mathscr{A} = (A, L, l_0, C, I, T)$ に対して上記の意味づけのもとで，初期状態から始まる，*(*状態数無限の*)* 状態遷移系を定義できる.
これを $\mathscr{T}(\mathscr{A})$ で表記する.

一般に確率時間オートマトンを考えるときは, ゼノンのパラドックスを回避するため以下の性質 (Non-Zeno) を仮定する.

定義 2.8 (Non-Zeno)**.** 有限の時間内に発生する離散遷移は有限個である.

2.7.2 確率時間オートマトンの状態空間の表現

確率時間オートマトンの無限の状態空間の表現方法として，ゾーンと呼ばれる表現を用いている．ゾーンはクロックに関する制約の集合であり，ゾーンが持つ制約を満たす全ての状態を表現する．確率時間オートマトンが持つ状態空間は有限個のゾーンから構成されるゾーングラフで表現することができる．また，ゾーンが持つ制約の集合は，DBM(Difference Bound Matrix) とよばれる行列として表現することができる [2][9]．DBM では，時間制約が一般に 2 クロック変数の差分不等式の集合で表されることを利用し，確率時間オートマ

トンが持つ n 個のクロックに対して $(n+1) \times (n+1)$ の行列として全ての制約を表現する．DBM のクロック制約を満たす全ての解の空間を D とすると，時間オートマトンの状態集合として $(l, D) = \cup_{u \in D}(l, u)$ として表現することとする．また，全ての $l \in L$ に対して $D_l^{inv} = \cup_{u \in I(l)} u$ と表現する．

2.7.3 Clock Zone と DBM

本質的にクロックの値が非負であることに注意するとクロック制約式は以下の形で表すことができる．

$x_0 = 0 \bigwedge_{0 \leq i \neq j \leq n} x_i - x_j \sim c_{i,j}$

ここで $\sim$ は $\leq$ または $<$, $c_{i,j}$ は非負実数である. また x_0 は一クロックの制約式も 2 クロックの差分不等式として表せるよう 0 定数を表すために便宜上導入されている．

この形の表現式を Clock Zone と呼ぶ.

さらに, Clock Zone のかわりに，クロック数が n のときに $n+1 \times n+1$ の行列を考え，この行列の要素として $c_{i,j}, \sim$ の 2 項組を与えると DBM となる．

クロックゾーンに関して以下の操作を定義することができる．

Intersection ϕ, ψ がクロックゾーンのとき $\phi \wedge \psi$(意味はこれらの論理積) もクロックゾーンである (クロックゾーンの式の定義より自明).

Projection ϕ がクロックゾーンでクロック変数 x が ϕ において自由のとき $\exists x \phi$ もクロックゾーンである．

Clock Reset $\mathcal{R}$ をクロックの集合とする. ϕ がクロックゾーンのとき $\phi[\mathcal{R} := 0]$(意味 ϕ 中のクロックのうち $\mathcal{R}$ に現れるクロックの値を 0 にする) もクロックゾーンである．
$\phi[\mathcal{R} := 0]$ は $\exists x_i(\phi \wedge \bigwedge_{x_i \in \mathcal{R}} x = 0)$ と等価である．

Elapsing Time ϕ がクロックゾーンで t を新しいクロック変数とするとき $\phi + t$ もクロックゾーンである．

ϕ がクロックゾーンのとき $\phi^{\Uparrow}$ もクロックゾーンである．

$\phi^{\Uparrow}$ は $\exists t \leq 0(\phi + t)$, ただし t は新しいクロック変数と等価である．

2.8 確率時間オートマトン

2.8.1 確率時間システム

定義 2.9 (確率時間システムの構文)**.** 確率時間オートマトン PTA の意味論を示す確率時間システム $\mathsf{TPS}_{\mathsf{PTA}}$ は，ラベル付けされたマルコフ決定過程であり，以下の *4* 項組 $\mathsf{TPS}_{PTA} = (S, s_0, TSteps, \mathcal{L})$ によって定義される．

- S：状態の有限集合
- s_0：初期状態
- $TSteps \subseteq S \times \mathbb{R}_0 \times \mathsf{Dist}(S)$：確率時間遷移関係 $(s, t, \mu) \in TSteps$ かつ $t > 0$ である場合，μ は点分布である．
- $\mathcal{L} : S \to 2^{AP}$：状態に原子命題式を割り当てるラベル付け関数

タプル (s, t, μ) の t は，$duration$ と呼ばれる．確率的なシステムと同様に，確率時間システムに対するアドバーサリとパスを導入する．

2.8.2 確率時間オートマトン

定義 2.10 (確率時間オートマトンの構文)**.** 確率時間オートマトン PTA は，*7* 項目 $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ によって定義される．

- L：ロケーションの有限集合
- l_0：初期ロケーション
- Act：アクションの有限集合
- $\mathcal{X}$：クロック変数の有限集合
- $inv : L \to Zones(\mathcal{X})$：インバリアントを割り当てる関数
- $prob \subseteq L \times Zones(\mathcal{X}) \times \mathsf{Dist}(2^{\mathcal{X}} \times L)$：確率エッジ関係をを割り当てる有限集合
- $\mathcal{L} : L \to 2^{AP}$：ロケーションへ原子命題式を割り当てるラベル付け関数

確率時間オートマトンの具体例を，図 2 に示す．

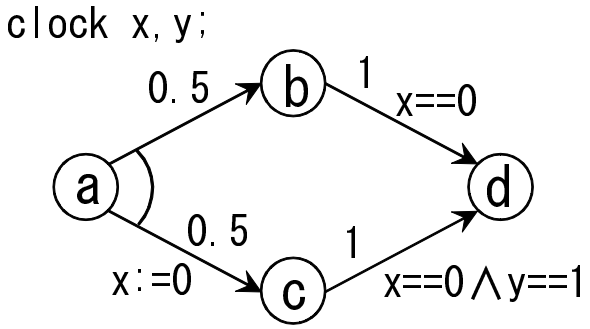

図 2: 確率時間オートマトンの例

定義 2.11 (確率時間オートマトンの遷移)**.** $(l, p, g) \in prob$ によって生成される確率時間オートマトンの遷移は，$p(X, l') > 0$ である 5 項目 (l, g, p, X, l') である．$\mathsf{edges}(l, g, p)$ を (l, g, p) によって生成される遷移の集合とする．また，$\mathsf{edges} = \{\mathsf{edges}(l, p, g) \mid (l, g, p) \in prob\}$ とする．

定義 2.12 (確率時間オートマトンの意味)**.** 確率時間オートマトン $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ の意味は，時間確率システム $\mathsf{TPS}_{\mathsf{PTA}} = (S, s_0, TSteps, \mathcal{L}')$ によって以下のように定義される．

- $S \subseteq L \times \mathbb{R}^{\mathcal{X}}$，$(l, \nu) \in S$ かつその時に限り，$\nu \triangleright inv(l)$ である．
- $s_0 = (l_0, \nu_0)$
- $((l, \nu), t, \mu) \in TSteps$ の場合，かつその時に限り，以下に従う．

  time transitions $t \leqslant 0$ の時，$\mu = \mu_{(l,\nu+t)}$ である．また，すべての $0 \geqslant t' \geqslant t$ に対し，$\nu + t' \triangleright inv(l)$ である．

  discrete transitions $t = 0$ であり，$\nu \triangleright g$，すべての $(X, l') \in \mathsf{support}(p)$ に対し，$\nu[X := 0] \triangleright inv(l')$ である $(l, g, p) \in prob$ が存在する場合，すべての $(l', \nu) \in S$ に対し，

$$\mu = \sum_{X \subseteq \mathcal{X} \& \nu' = \nu[X:=0]} p(X, l')$$

- $\mathcal{L}'((l, \nu)) = L(l)$ *for all* $(l, \nu) \in S$

2.8.3 $well\text{-}formed$ な確率時間オートマトン

確率時間オートマトンにおける確率的な遷移が常に有効である場合に限り，その確率時間オートマトンを $well\text{-}formed$ であるとする [22]．形式的には，確率時間オートマトン $\mathsf{PTA} = (L, l_0, Act, \mathcal{X}, inv, prob, \mathcal{L})$ が，常に以下の条件を満たせば well-formed である．

$$\forall (l, g, p) \in prob.\ \forall \nu \in \mathbb{R}^{\mathcal{X}}_{\geqslant 0}.\ (\nu \triangleright g)\ \rightarrow (\forall (X', l') \in \mathsf{support}(p).\ \nu[X := 0] \triangleright inv(l')).$$

確率時間オートマトンは，確率的な各遷移 $(l, g, p) \in prob$ におけるガード式を置き換えることで，単純に $well\text{-}formed$ な確率時間オートマトンに変換可能である．

$$(\wedge_{(X,l') \in \mathsf{support}(p)} [X := 0] inv(l')) \wedge g$$

この変換において，オートマトンの意味への影響は全く無い．本研究で扱う確率時間オートマトンは，すべて $well\text{-}formed$ な確率時間オートマトンであるとする．

2.9 確率時間オートマトンの同時実行可能性

確率時間オートマトンでは，ある性質を満たす確率で構成されるパスが複数個ある場合が存在する．この時，同じロケーションを出発するパスは，同じゾーンによって遷移している必要がある．確率時間オートマトンにおける時間遷移は，非決定的な遷移であり，異なるゾーンで各パスが出発している場合，それらのパスは異なるアドバーサリで動作しているためである．

ここで，同時実行可能性に関する次の補題を与える．

補題 2.1 (2 つのパスの同時実行可能性)**.** 確率時間オートマトン PTA 上の *2* つの任意のパス $\omega^\alpha = \langle l_0^\alpha, \nu_0^\beta \rangle \xrightarrow{p_0^\alpha, t_0^\alpha} \langle l_1^\alpha, \nu_1^\alpha \rangle \xrightarrow{p_1^\alpha, t_1^\alpha} \ldots \xrightarrow{p_{n-1}^\alpha, t_{n-1}^\alpha} \langle l_n^\alpha, \nu_n^\alpha \rangle$ と $\omega^\beta = \langle l_0^\beta, \nu_0^\beta \rangle \xrightarrow{p_0^\beta, t_0^\beta} \langle l_1^\beta, \nu_1^\beta \rangle \xrightarrow{p_1^\beta, t_1^\beta} \ldots \xrightarrow{p_{m-1}^\beta, t_{m-1}^\beta} \langle l_m^\beta, \nu_m^\beta \rangle$ が以下の性質を満たすとき，ω^α と ω^β は同時実行可能である．

$\mathsf{isCompatible}(\omega^\alpha, \omega^\beta) =$

$$\begin{cases} true & \text{if}(\ \exists i \in \mathbb{N} \text{ such that } \omega^\alpha(i+1) \neq \omega^\beta(i+1) \\ & \wedge \forall j \leqslant i \in \mathbb{N}.\ \omega^\alpha(j) = \omega^\beta(j)).\ t_i^\alpha = t_i^\beta \\ false & otherwise \end{cases}$$

true になる条件としては，パスにおける i 番目の状態で同じ状態のものの中で，そのロケーションにおける時間経過が同じものである．さらに，$i+1$ 番目の状態が異なるようなパスである．単純には，i 番目まで全く同一の遷移が行われ，$i+1$ 番目の遷移によって異なる遷

移が行われるものである．さらに，この補題を元に，3 つ以上のパス集合に関する同時実行可能性についての補題を与える．

補題 2.2 (2 つ以上のパスの同時実行可能性)**.** 確率時間オートマトン PTA 上のパス集合 Ω が以下の性質を満たすとき，Ω は同時実行可能である．

$\mathsf{isCompatible}(\Omega) =$

$$
\begin{cases}
true & \text{if}(\ \exists i \in \mathbb{N} \text{ such that } \bigwedge\limits_{\substack{\omega^\alpha,\omega^\beta \in \Omega \\ \wedge \omega^\alpha \neq \omega^\beta}} \omega^\alpha(i+1) \neq \omega^\beta(i+1) \\
& \wedge (\forall j \leqslant i \in \mathbb{N}. \bigwedge\limits_{\substack{\omega^\alpha,\omega^\beta \in \Omega \\ \wedge \omega^\alpha \neq \omega^\beta}} \omega^\alpha(j) = \omega^\beta(j)). \\
& \bigwedge\limits_{\substack{\omega^\alpha,\omega^\beta \in \Omega \\ \wedge \omega^\alpha \neq \omega^\beta}} t_i^\alpha = t_i^\beta \\
& \wedge (\forall \Omega' \in 2^\Omega \text{ such that } \Omega' \neq \Omega \wedge |\Omega'| \geq 2). \\
& \mathsf{isCompatible}(\Omega') \\
false & otherwise
\end{cases}
$$

補題 2.1 では 2 つの場合を与えたが，補題 2.2 ではパスの集合 Ω に関するものである．$\mathsf{isCompatible}(\Omega)$ が true になる条件は，集合に含まれる任意の 2 つのパスが同時実行可能であり，かつその部分集合も同様である，というものである．

続いて，同時実行可能性を満たさないようなパスに関する例を与える．図 2 におけるロケーション a から d へ遷移するパスを求めた場合，ロケーション b を経由して d に遷移するパスは，$\omega^\alpha = \langle a, x = 0 \wedge y = 0\rangle \xrightarrow{0.5,0} \langle b, x = 0 \wedge y = 0\rangle \xrightarrow{1.0,0} \langle d, x = 0 \wedge y = 0\rangle$，となり，ロケーション c を経由して d に遷移するパスは，$\omega^\beta = \langle a, x = 0 \wedge y = 0\rangle \xrightarrow{0.5,1} \langle c, x = 0 \wedge y = 1\rangle \xrightarrow{1.0,0} \langle d, x = 0 \wedge y = 1\rangle$，となる．$\omega^\alpha$ で d へ遷移するためには，$b \to d$ の遷移に対するガード式 $x == 0$ が存在するため，a において時間遷移無しで動作する必要がある．一方，ω^β で d へ遷移するためには，$b \to d$ の遷移に対するガード式 $x == 0 \wedge y == 1$ が存在するため，a において 1 単位時間の遷移を行い，クロック変数 y を a の時点でインクリメントしておかなければ d へ到達できない．

図 2 の確率時間オートマトンのようにロケーションにおける離散遷移の分岐において出発条件が異なる場合，各パスの時間的な振る舞いを考慮して反例となるパス群を考慮する必要がある．

2.10 CEGAR loop

CEGAR ループ [7] とは，CounterExample-Guided Abstraction Refinement の略であり，直訳すると反例に基づいた抽象化洗練ループである．図 3 に，一般的な CEGAR フレームワー

クを示す.

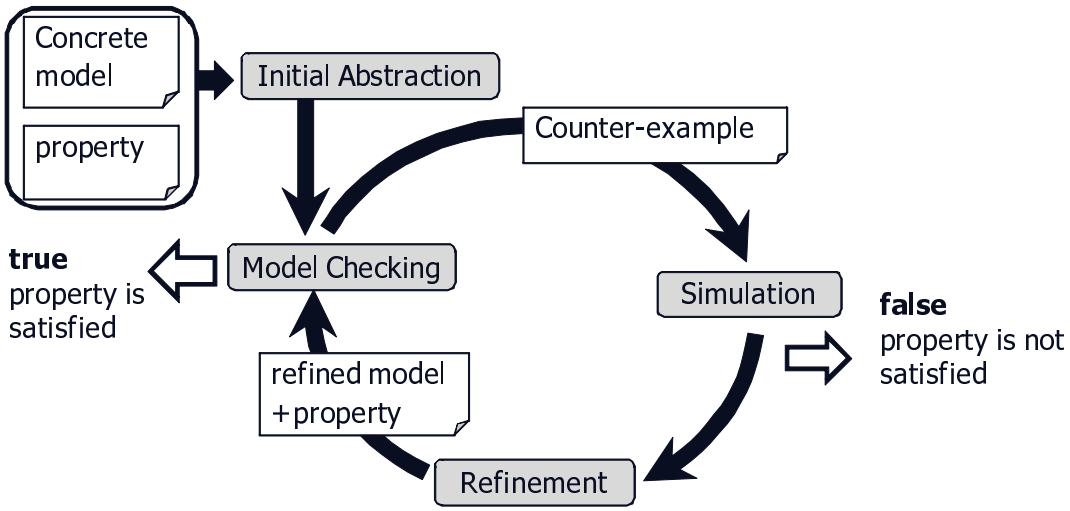

図 3: CEGAR ループ

モデルの抽象化によって，本来の抽象化前のモデル（具体モデル）上で発見され得ない，偽反例が発見されてしまうことがある．一般的な CEGAR ループでは，抽象化済みモデル（抽象モデル）上で発見された偽反例を具体モデル上でのシミュレーションによって確認する．この時，具体モデル上で起こり得ない反例であった場合は，抽象モデル上でその反例が二度と発見されないようにモデルの改善（洗練）を行う．この繰り返しによって，最終的に具体モデル上で発生するであろう反例を見つけ，検査を終了する．

ただし，最初のモデル検査の段階で反例が出力されなければ，over approximation を満たす抽象化に限定されているため，具体モデル上でも反例が必ず存在しない．よって，この場合は検査結果を true として終了する．

2.11　時間オートマトンの抽象化洗練手法

著者らは [25][26] で，Clarke らの Counter-example Guided Abstraction Refinement[7] の枠組みを利用した，時間オートマトンに関する抽象化洗練手法を提案している．この手法では，時間に関する制約をすべて除去する抽象化行い，抽象モデル上で偽反例が出力された場合，抽象状態の複製によって抽象モデルの洗練を行う．

3　関連研究

本研究では，確率時間オートマトンに対するモデル検査手法を提案している．
ここでは，既存の確率時間オートマトンに対するモデル検査アプローチを簡潔に紹介する．

3.1　forward Reachability

forward Reachability アルゴリズム [20] は，確率時間オートマトンを前方に探索してゾーングラフ化することで，初期状態から到達可能なゾーングラフ（Reachability graph）を生成する．
そのグラフに対し，マルコフ決定過程と同様の到達可能性解析を実行することで，確率時間オートマトンに対する到達可能性解析を行う．
このアプローチは，DBM のようなデータ構造と共に実行することで，極めて効率的な方法であるため，有効な手法である．しかしながら，確率時間オートマトンの同時実行可能性問題を解決できないため，この技術は最大到達確率の上限値を求めるに留まっている．

3.2　backward Reachability

backward Reachability アルゴリズム [22] もその名の通り，forward Reachability アルゴリズムとは全く逆の手法で，性質を満たす集合から初期状態まで，後ろ向きに探索を行っていく手法である．
性質を満たす状態集合を生成し，その集合とモデルを元に，バックワードに状態を拡大していく．拡大された状態が初期状態まで到達した場合，そのグラフをマルコフ決定過程と同様の到達可能性解析を実行し，モデル検査を行う．この手法は，最大，最小到達確率を正確な値で計算できるが，計算コストが非常に高いために，適用範囲に限界がある．

3.3　Digital clocks[19]

確率オートマトン上に，Integer の制約としてクロック変数を記述したものである（図 4）．Integer の制約なので，本来は実数であるべきクロック変数を，自然数として記述することになる．この手法では，確率オートマトンを記述する形になるので計算は容易だが，状態数は爆発的に増加し，さらに，自然数として記述することで，元の確率時間オートマトン上で存在する状態が一部削られてしまう．よって，検査を実行する性質によっては，誤った結果を得ることになる．結局，限られたクラスの確率時間オートマトンに対してしか適用することができない．

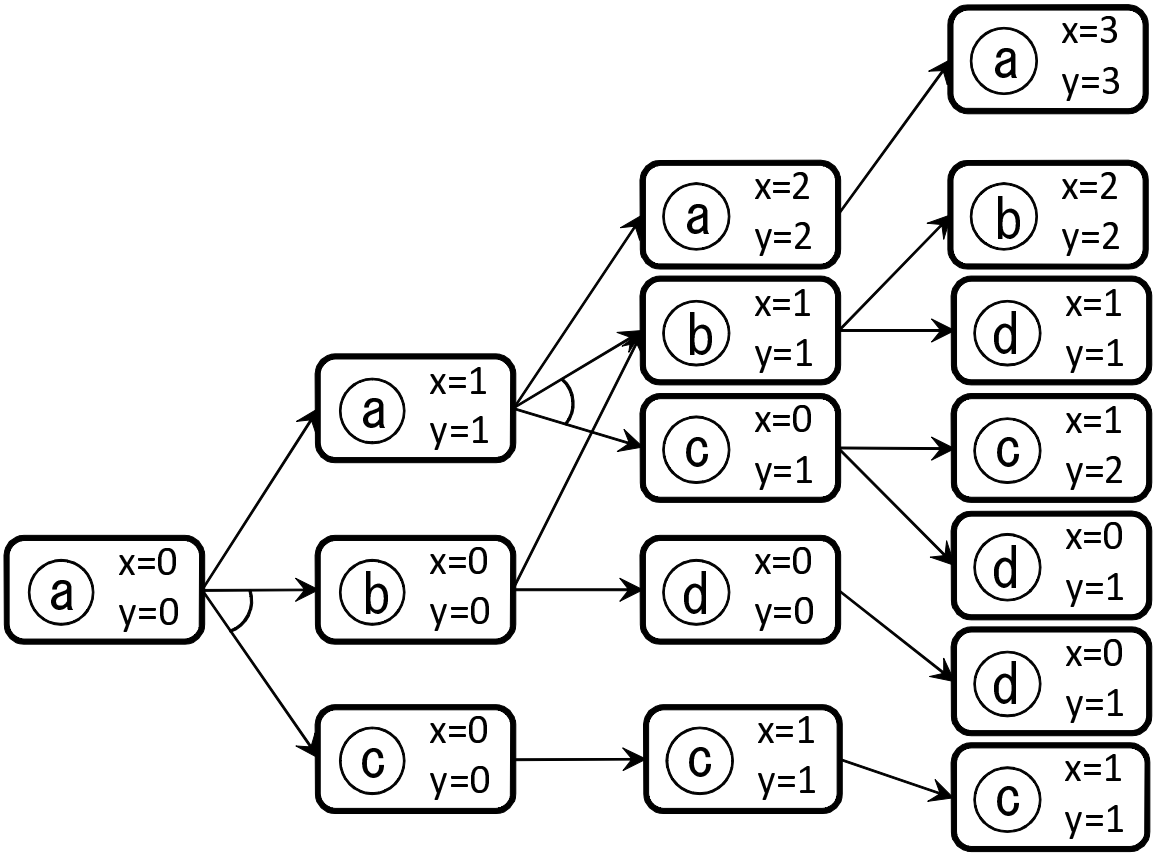

図 4: Digital clocks で表した確率時間オートマトン

3.4 Stochastic game based

上記 3 つの手法を踏まえて考案された技術がこれから示す Stochastic game verification[18] である．この手法では，forward Reachability アルゴリズムによって得られたグラフから，確率ゲームの形式に変換・修正を実行することによってグラフの洗練を繰り返していく手法である．

具体的な手法の流れは，図 5 によって示される．

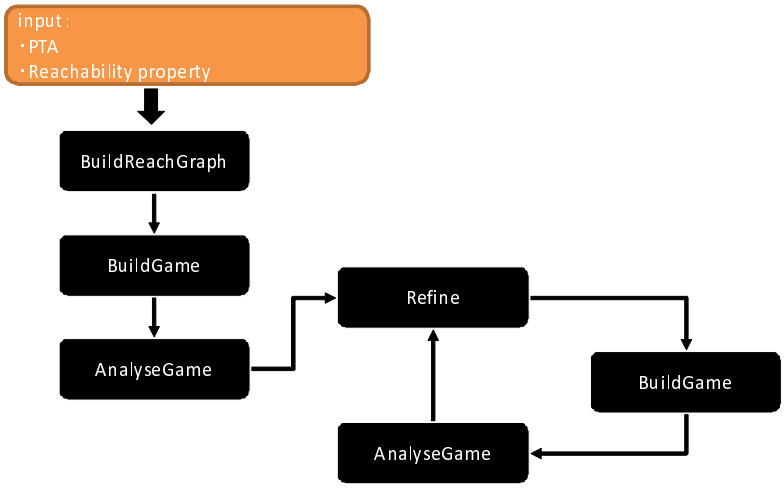

図 5: Stochastic games for Verification 概要

3.5 その他の手法

その他の手法として，時間オートマトンモデル検査ツール KRONOS[11] と，PRISM を組み合わせて解析を行う手法が提案されている [10].

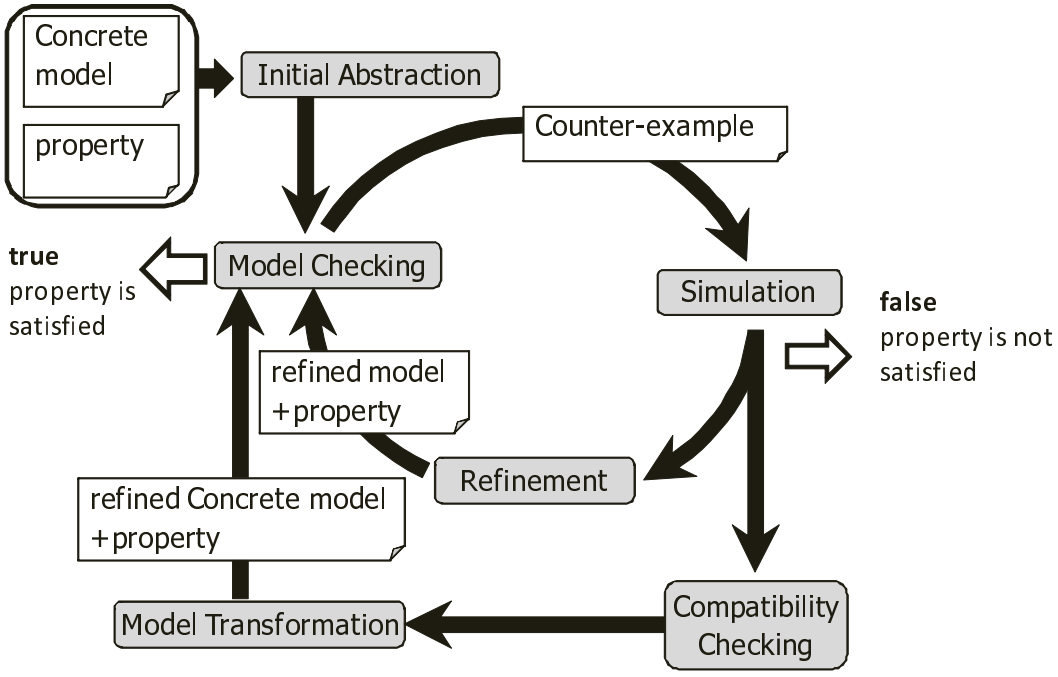

図 6: 提案手法

4 提案手法

本章では，本稿で提案する確率時間オートマトンの抽象化洗練手法を示す．提案する抽象化洗練手法では，著者らが [25][26] で提案した時間オートマトンの抽象化洗練手法を利用している．さらに，前節で述べた，同時実行可能性を考慮した反例を生成するために，後方シミュレーションに加えて，時間遷移を明確化したロケーションを生成することで，同時実行可能性を明確化した確率時間オートマトンを生成する．

図 6 は本稿で提案する抽象化洗練の枠組みである．図で示すように，同時実行可能性を明確化するため，一般的な CEGAR の枠組みにモデル変換のフローを追加している．入力として，確率時間オートマトン PTA，性質を与える．検査対象とする性質は，$P_{<p}\,[\,true\,U\,\texttt{err}]$ という PCTL 式に限定する．

これは，err（エラー状態を示す）に到達する確率は p 以下である，という到達可能性解析に関するモデル検査を実行する式である．err には，PCTL の状態式に関する任意の記述を与える．

4.1 初期抽象化

初期抽象化では，[25][26] と同様に，クロック変数に関する制約をすべて除去することで，over approximation を満たすように抽象化を行う．

確率時間オートマトンのクロック変数に関する制約をすべて除去するため，この初期抽象化では，確率時間オートマトンをマルコフ決定過程に変換することになる．よって，後述のモデル検査では，マルコフ決定過程の既存の検査手法を採用し，マルコフ決定過程における到達可能性解析を実行する．

4.2 モデル検査

モデル検査では，抽象モデルとして生成されたマルコフ決定過程に対して ValueIteration を適用する．

本手法では，到達可能性の最大確率を計算する手法を提案しているため，ValueIteration に入力するパラメータを，$f = max$，$g = max$ に設定してモデル検査を実行する．このアルゴリズムによって計算された結果が入力された性質の p を満たしている場合，確率時間オートマトンのモデル検査結果を $true$ として計算を終える．

ValueIteration では，到達可能性の最大確率を計算することが可能であるが，入力された性質の p を上回る確率を持つ場合の具体的なパス（群）を示さない．よって，抽象モデル上で反例となるパス（群）を探索するために，文献 [1] で提案されている，PCTL 式に対する反例を探索する手法を採用する．

4.3 シミュレーション

シミュレーションでは，k-最短路探索によって得られたパス（群）に対して，元の確率時間オートマトン上で実行可能かどうかを調べる．本手法では，[26] で提案されているシュミレーションアルゴリズムに従って，DBM の演算によって到達可能か判定する．

4.4 抽象モデルの洗練

抽象モデル上で発生した偽反例を，元の確率時間オートマトン上で発生しないように，抽象モデルの洗練を行う．本手法では，[26] で提案されている抽象状態の複製による手法を用いる．

4.5 同時実行可能性の検査

k-最短路探索によって得られたパス（群）が全て元の確率時間オートマトン上で実行可能であり，かつ確率の和が与えられた確率 p を超える場合，さらにそれらのパスの同時実行可能性（補題 2.2）を調べる必要がある．確率時間オートマトンのある性質に対する反例中に同時実行可能性が存在する場合，その反例は正しくない．

そのために，反例として出力されたパス群に対し，同時実行可能性が偽となる分岐が存在しているかどうかを判定する．まず後方シミュレーションを行って，各パスを通過する各ロケーションにおいて目的ロケーションへ到達するために必要となる出発条件を求める．その後，補題 2.2 に従って，各分岐に対して同時実行可能かどうかを検査する．

4.5.1 後方シミュレーション

後方シミュレーション（Algorithm2）では 1 つのパスに対し，そのパスが通過するすべてのロケーションに対する出発条件を求める．

離散遷移 $e = (l_{prev}, g_e, p_e, X_e, l)$ に対してロケーション l から l_{prev} への後方シミュレーションを考える．

まず，後方シミュレーションを行った際に，リセット式によって除去される制約式を求める関数を以下のように定義する．入力となる ζ は，$inv(l)$ である．

$$\mathsf{free}(\zeta, e) = \begin{cases} \mathsf{free}(\zeta^{\alpha}, e) \wedge \mathsf{free}(\zeta^{\beta}, e), & \text{if } \zeta = \zeta^{\alpha} \wedge \zeta^{\beta} \\ true, & \text{if } \zeta = x \sim c \text{ where } x \in X_e \\ & \text{ or } \zeta = x - y \sim c \text{ where } x \in X_e \text{ or } y \in X_e \\ \zeta, & \text{otherwise} \end{cases} \tag{5}$$

$\mathsf{free}(\zeta, e)$ では，入力されたゾーン中に，遷移 e 上のリセット式 X_e に含まれるクロック変数に関する制約が含まれていた場合，その制約を除去する，という計算を行っている．

free を用いてロケーション l_{prev} から l を示すエッジ e 上で後方シミュレーションを実行し，l_{prev} から l に出発条件となる遷移可能なゾーンを求めると，以下のように示される．

$$\zeta^{l_{prev}} = \mathsf{free}(inv(l), e) \wedge g_e \wedge inv(l_{prev})$$

ここで，$\zeta^{l_{prev}}$ を l_{prev} における出発条件として決定する．つづいて，$l_{prev} = l_0$ でなければ，続けて後方シミュレーションを行う．この時，$\zeta^{l_{prev}}$ に対し，時間経過の逆算である down 演算を行い，先ほどの式を用いてシミュレーションを継続していく．down 演算を行う関数を以下のように定義する．

$$\mathsf{down}(\zeta) = \begin{cases} \mathsf{down}(\zeta^{\alpha}) \wedge \mathsf{down}(\zeta^{\beta}), & \text{if } \zeta = \zeta^{\alpha} \wedge \zeta^{\beta} \\ true, & \text{if } \zeta = x \geqslant c \\ \zeta, & \text{otherwise} \end{cases} \tag{6}$$

$\mathsf{down}(\zeta)$ では，入力されたゾーン中の下限値に関する制約についてすべて除去する，という計算を行っている．この演算によって,「時間経過後のゾーン」から,「時間経過前から時間経過後のゾーン」に拡大することができる．

以上の流れを用いて，パス ω に対する後方シミュレーションのアルゴリズムを Algorithm2 に示す．入力には，元の確率時間オートマトン PTA と，反例候補 $\omega = l_0 \xrightarrow{p_1} l_1 \xrightarrow{p_2} \cdots \xrightarrow{p_n} l_n$ を用いる．

Algorithm 2 BackwardSimulation(PTA, ω)

1: $\zeta := true$
2: **for** $i := n$ down to 2 **do**
3: $\{n$ はパス ω の長さ $\}$
4: $l_{curr} := \omega[i], l_{prev} := \omega[i-1]$
5: $\zeta := \zeta \wedge inv(l_{curr})$
6: $e := (l_{prev}, g, p, X, l_{curr})$ $\{\in \mathsf{edges}\}$
7: $\zeta := \mathsf{free}(\zeta, e) \wedge g_e \wedge inv(l_{prev})$
8: $\mathsf{Z}_{\omega}[i] := \zeta$ $\{\mathsf{Z}_{\omega}[i]$ に出発条件を追加 $\}$
9: $\mathsf{ER} := \mathsf{ER} \cup (l, l_{curr}, e, \zeta)\{$ 出発条件と遷移の関係を保持 $\}$
10: $\mathsf{down}(\zeta)$
11: **end for**
12: **return** $(\mathsf{Z}, \mathsf{ER})$

このアルゴリズムでは，得られたパスを後方にシミュレーションを行い，$\zeta^{l_{prev}}$ に対しゾーンを求めていく．

後方シミュレーションが適用されるとき，与えられるパスは既に (前方) シミュレーションによって実行可能であると判定されているため，後方シミュレーションにおけるゾーンの計算の中で空になることはない．

パス中の各ロケーションにおいて求められた出発条件 ζ を，$\mathsf{Z}^{l}_{\mathsf{path}}[l_{prev}]$ に追加しておく．

上記の条件を満たし，$l_{prev} = l_0$ であれば，シミュレーションを終える．そうでなければ $\zeta^{l_{prev}}$ に対し，時間経過の逆算を行い，ゾーンを拡大する．その後，残りの経路に基づいて上述のゾーン演算を実行していく．

Algorithm 3 IsCompatible($\Omega, \mathsf{Z}, index$)

$\zeta := true, \mathsf{NSIM} := \emptyset, NLS := \emptyset$
for all $\omega' \in \Omega$ such that $|\omega'| > index$ **do**
  $l_{next} := \omega'[index + 1]$
  if $l_{next} \notin NLS$ **then**
    $NLS := NLS \cup l_{next}$
    $\Omega_{l_{next}} := \emptyset$
  end if
  $\Omega_{l_{next}} := \Omega_{l_{next}} \cup \{\omega'\}$
end for
if $|NLS| \geqslant 2$ **then**
  for all $\omega' \in \Omega$ **do**
    $\zeta := \zeta \wedge \mathsf{Z}_{\omega'}[index]$
    if $\zeta = false$ **then**
      $\mathsf{NSIM} := \mathsf{NSIM} \cup \{\omega'[index]\}$ { 同時実行できないロケーション集合 NSIM を作成 }
      break
    end if
  end for
  for all $l_{next} \in NLS$ **do**
    $\mathsf{NSIM} := \mathsf{NSIM} \cup \text{IsCompatible}(index + 1, \Omega_{l_{next}})$
  end for
end if
return NSIM

4.5.2 同時実行可能性の判定

後方シミュレーションによって各パス，各ロケーションの出発条件が求められたが，同時実行可能性の補題における isCompatible を満たしているかを調べる必要がある．そのために，IsCompatible（Algorithm3）を実行する．

このアルゴリズムでは，後方シミュレーションによって求められた出発条件その条件を持つパスの集合を元に，同時実行性を検査する．最終的に，NSIM に同時実行できないロケーションの集合が求められる．

具体的には，抽象モデル上のパスであるロケーションの系列 ω の集合 Ω が与えられ，Ω に含まれるパスの分岐が存在するかどうかを調べる．その分岐上で出発条件を比較し，同時に実行できないと判断された場合，同時実行できないロケーションの集合 NSIM に追加される．分岐において同じ遷移を示すパスは，その集合を用いて再帰的に IsCompatible を実行し，パス中のすべてのロケーションに対する同時実行可能性を検査する．

4.6 同時実行可能性を考慮したモデル変換

同時実行可能性の検査によって，同時実行可能性が偽であると判断された場合，偽であると判定された分岐において，時間遷移による振る舞いの違いを明確にするために，新たなロケーションを生成し，遷移関係を操作する．この複製されるロケーションを，本稿では複製ゾーンロケーションと呼ぶ．

直観的には，あるロケーション l 上の時間遷移が取り得るすべてのパターンを，離散遷移上で明示的に行うようにする．この時間遷移が取れるパターンに応じたロケーションの複製を行って，元モデル PTA に対し変換を施す．同時実行可能性となる反例パスが出力されてしまうのは，PTA 上の時間制約に関するラベルをすべて除去してしまうため，非決定的な時間遷移について同じパス群に含まれてしまうためである．このようなケースに対し，時間遷移を離散遷移のように扱うことで抽象モデル上で異なる非決定的な遷移として扱うようになり，抽象モデル上でも同時に実行可能でない，と判断できる．

変換の手順として，具体的には次のように処理が実行される．

Algorithm3 によって得られた集合 NSIM の要素であるロケーション l に対し，l において出発条件が異なるパスが存在する，つまり同時に実行不可能な分岐であると判断した場合，TransformPTA(Algorithm4) によって複製ゾーンロケーションの生成を実行する．ただし，TransformPTA に入力される確率時間オートマトン PTA は，時間に関する制約のラベルを除去した抽象モデル MDP において，各状態において取りうるアクションが（ValueIteration によって）決定されている PTA とする．

まず，複製されるロケーションのインバリアントを示す．元モデル $\mathcal{M}$ のロケーション l

に対応する，複製ゾーンロケーション $\tilde{l}$ における $inv(\tilde{l})$ を示す．

以下のように複製ゾーンロケーションのインバリアント $inv(\tilde{l})$ を求める関数 dupinv を定義する．

$$\begin{aligned} & \mathsf{dupinv}(l, l_{prev}, e, \zeta) \\ = \quad & \nearrow ((\neg\zeta \wedge inv(l_{prev}) \wedge g_e)[X := 0]) \wedge inv(l) \end{aligned}$$

入力となる l は複製元のロケーション，l_{prev} は複製元のロケーションへ 1 ステップで遷移できるロケーションである．また，e はその遷移を示すエッジである．ζ は，l を経由して目的の状態へ到達するために必要となる出発条件である．この処理を行うために，後方シミュレーションで得られた出発条件と，その出発条件がどのロケーションに対する出発条件なのかについての関係の情報を保持しておく必要がある．そのために，後方シミュレーションアルゴリズムの 19 行目において，出発上限と遷移の関係を示す集合 ER を定義しておき，dupinv に対する入力を与える（アルゴリズム 6，4 行目）．

Algorithm 4 TransformPTA(PTA, Z, a)

1: DLS := $\emptyset$, POWZ := $\emptyset$
2: **for all** $l \in$ NSIM **do**
3: POWZ := PowZ(l, Z)
4: E :=getTransitionSet(l) {E は l における遷移の集合 }
5: DLS := MakeDuplicateZoneLocation(l, DLS)
6: MakeTransition(l, a, POWZ, DLS)
7: RemoveTransition(E) { 元の遷移 E の除去 }
8: **end for**
9: **return** PTA

続いて，複製ゾーンロケーション $\tilde{l}$ の遷移関係について述べる．まず，$\tilde{l}$ からの遷移は，複製元ロケーションからの遷移をそのまま保持する．ただし，$inv(\tilde{l})$ を満たすガードの遷移のみとする．この処理はアルゴリズム 6 の 6 行目に相当する．

複製ゾーンロケーションを追加した場合，そのロケーションへの遷移だけでなく，出発条件の組み合わせに応じた遷移を追加する．例えば，出発条件が ζ_1, ζ_2 の 2 つのパターンが発見された場合，$\zeta_1 \wedge \zeta_2$，$\neg\zeta_1 \wedge \zeta_2$，$\zeta_1 \wedge \neg\zeta_2$，$\neg\zeta_1 \wedge \neg\zeta_2$ の 4 つのパターンが存在すると考えられる．今回の提案手法では，そのパターンを明確に区別し，そのパターンに応じた遷移を追加することによって，出発条件の異なる，つまり同時実行不可能な遷移を厳密に分解する．ただし，空となるパターンは区別せず，遷移の追加を行わない．

提案するアルゴリズム 7 では，出発条件のパターンを用いて，複製ゾーンロケーションへ

Algorithm 5 PowZ(l, ER)

1: $\zeta_{pow} := true, Z_{oc} := \emptyset, \mathsf{POWZ[l]} := \emptyset$
2: **for all** $(l_{curr}, l', e, \zeta) \in \mathsf{ER}$ such that $l_{curr} = l$ **do**
3: $Z_{oc} := Z_{oc} \cup \{\zeta\}$ {l における出発条件の集合を取得 }
4: **end for**
5: **for all** $Z \in \mathsf{POWERSET}(Z_{oc})$ **do**
6: {POWERSET によってべき集合を作成し，要素 Z を取りだす }
7: **for all** $\zeta^l \in Z_{oc}$ **do**
8: **if** $\zeta^l \in Z$ **then**
9: $\zeta_{pow} := \zeta_{pow} \wedge \zeta^l$
10: **else**
11: $\zeta_{pow} := \zeta_{pow} \wedge \neg\zeta^l$
12: **end if**
13: **end for**
14: **if** $\zeta_{pow} \neq \emptyset$ **then**
15: $\mathsf{POWZ}[l] := \mathsf{POWZ}[l] \cup \{\zeta_{pow}\}$
16: **end if**
17: $\zeta_{pow} := true$
18: **end for**
19: **return** POWZ[l]

の遷移，元の PTA からそのまま保持するロケーションへの遷移，の 2 つの遷移生成アルゴリズムを示している．それぞれ，6，13 行目によって遷移の追加処理が行われる．

例えば，powZ によって生成された l における出発条件のパターンのうち $\zeta_1 \wedge \neg\zeta_2$ についての遷移関係を構築する場合，l からの遷移は，ζ_1 の出発条件で遷移できる次のロケーション (この場合 l_1 とする) は，そのまま追加し，ζ_2 は否定が付いているため，l_2 の複製ゾーンロケーションへ遷移することになる．この処理を繰り返すことで，出発条件のパターン毎の遷移関係を構築することができる．

4.7 確率時間オートマトンの変換例

図 2 の確率時間オートマトン上での複製ゾーンロケーションの生成例を以下に示す．

まず，前提として，すでに反例候補のパスが探索されているものとする．この PTA の抽象モデル上で探索されるパスは以下の 2 つ（ω_1，ω_2）である．

Algorithm 6 MakeDuplicateZoneLocation(l, DLS)

for all $l_{tmp} \in NEXT(l)$ **do**
  $l_{dup} := newLocation()$ { ロケーションを生成 }
  for all $(l_{prev}, l', e, \zeta_{tmp}) \in$ ER such that $l_{prev} = l$ and $l_{tmp} = l'$ **do**
    $inv(l_{dup}) :=$ dupinv$(l, l_{tmp}, e, \zeta_{tmp})$
  end for
  $L := L \cup \{l_{dup}\}$
  DLS := DLS $\cup \{l_{dup}\}$ { 複製ゾーンロケーションの集合を生成 }
  for all $(l_{prev}, g, p, X, l') \in$ edges such that $l_{prev} = l_{tmp}$ and $g \in inv(l_{dup})$ **do**
    edges := edges $\cup \{(l_{dup}, g, p, X, l')\}$
  end for
end for
return DLS

- $\omega_1 = a \rightarrow b \rightarrow d$
- $\omega_2 = a \rightarrow c \rightarrow d$

これらのパスは，シミュレーション行った結果，元の PTA 上でも到達可能である．これらのパスを元に，後方シミュレーションを行う．ω_1 では，a における出発条件が $x == 1 \wedge y == 0$（ζ_1 とする）となる．ω_1 では，a における出発条件が $y == 1$（ζ_2 とする）となる．また，算出された出発条件は 2 つであり，2 つの条件を組み合わせを考える（powZ の計算を行う）と，

- $\zeta_1 \wedge \zeta_2$
- $\zeta_1 \wedge \neg\zeta_2$
- $\neg\zeta_1 \wedge \zeta_2$
- $\neg\zeta_1 \wedge \neg\zeta_2$

となる．これらの条件より実際の出発条件を計算すると，

- $\zeta_1 \wedge \zeta_2 = \emptyset$
- $\zeta_1 \wedge \neg\zeta_2 = x == 0 \wedge 1 > y \leq 0 \wedge y > 1$
- $\neg\zeta_1 \wedge \zeta_2 = x > 0 \wedge y > 0$
- $\neg\zeta_1 \wedge \neg\zeta_2 = x > 0 \vee 1 > y > 0 \wedge y > 1$

Algorithm 7 MakeTransition($l, \mathsf{POWZ}[l], \mathsf{DLS}$)

for all $\zeta \in \mathsf{POWZ}[l]$ **do**
  for all $l_{next} \in NEXT(l)$ **do**
    $\{NEXT$ は l の次状態の集合を表す $\}$
    if $l_{next} \in \mathsf{DLS}$ **then**
      $l_{dup} := DupZoneMap(l_{next})$
      for all $(l_{prev}, g, p, X, l') \in \mathsf{edges}$ such that $l_{prev} = l$ **do**
        $\mathsf{edges} := \mathsf{edges} \cup \{(l_{prev}, g, p, X, l_{dup})\}$
      end for
    else
      for all $(l_{prev}, g, p, X, l') \in \mathsf{edges}$ such that $l_{prev} = l$ and $l' = l_{next}$ **do**
        $\mathsf{edges} := \mathsf{edges} \cup \{(l, g, p, X, l')\}$
      end for
    end if
  end for
end for

Algorithm 8 RemoveTransition(E)

$\mathsf{edges} := \mathsf{edges} \setminus E$

となり，実際に生成される遷移関係は図 7 のようになる．

次に，複製ゾーンロケーションの生成を行う．複製ゾーンロケーションは，ロケーション a から同じアクションによって遷移できる遷移先に対し生成される．この場合，ロケーション b, c に対して複製を行う．複製ゾーンロケーション b' では，b を通るパス ω_1 の，a での出発条件を用いて $inv(b')$ の計算を行う．よって，

$$inv(b') = x > 0 \vee y > 0$$

となる．c' では，c を通るパス ω_2 の，a での出発条件を用いて $inv(c')$ の計算を行う．

$$inv(b') = (y < 1 \vee y > 0) \wedge x == 0$$

となる．さらに，遷移関係を構築すると，$inv(b')$ では $b \xrightarrow{x==0} d$ を満たさないため，b' に対する遷移の追加は行われない．また，$inv(c')$ では $b \xrightarrow{x==0 \wedge y==1} d$ を満たさないため，c' に対する遷移の追加も行われない．よって，以下の図 8 のような確率時間オートマトンになる．

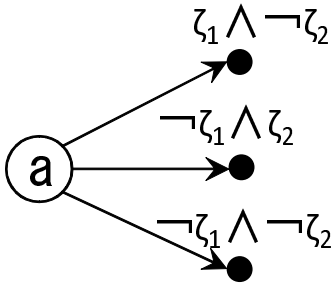

図 7: 出発条件に対する遷移関係

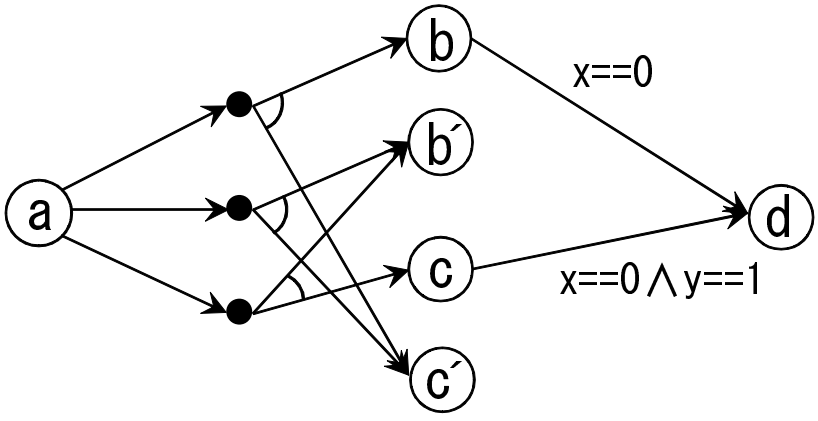

図 8: 変換後の確率時間オートマトン

4.8 モデルの等価性に関する証明

本稿で提案する抽象化洗練手法では，確率時間オートマトンのパス群に対する同時実行可能性を考慮し，各パスが各ロケーションにおいて，どのくらいの時間遷移を実行する必要があるか，を明確化するために複製ゾーンロケーションの生成を行う．

ここでは，前章において述べてきた変換後の確率時間オートマトンに対する定義を用いて，モデルの変換前後の等価性について証明を行う．

$\tilde{L}$ から複製元である L にマップする関数 $DupZoneMap$:$\tilde{L} \rightarrow L$ を $DupZoneMap(\tilde{l}) = l$ とする．このマップでは，入力として変換後のモデル $\mathcal{M}$ のロケーション $\tilde{l}$ を与えると，元モデル $\mathcal{M}$ の複製元に対応するロケーション l を返すものとする．

元モデル $\mathcal{M}$ の任意の状態を $\langle l, \nu \rangle$ とし，さらに変換後のモデル $\tilde{\mathcal{M}}$ から得られた任意の状態を $\langle \tilde{l}, \tilde{\nu} \rangle$ とする．また，$\mathcal{M}$ のロケーションの集合を L とし，$\tilde{\mathcal{M}}$ の複製ゾーンロケーションの集合を $\tilde{L}$ とする．

$\tilde{L}$ から複製元である L にマップする関数 $DupZoneMap$ を定義する．

$$DupZoneMap : \tilde{L} \rightarrow L$$

以下では，この複製ゾーンロケーションの生成前後において，モデルの等価性に関する証明を行う．

定理 4.1. 元の確率時間システム $\mathcal{M}$ と，変換後の確率時間システム $\tilde{\mathcal{M}}$ は等価である．

証明 4.1. 変換前後のモデルにおける各状態 $(\langle l, \nu \rangle, \langle \tilde{l}, \nu \rangle)$ の対応関係は，以下の R によって定義する．

$R = \{(\langle l, \nu \rangle, \langle \tilde{l}, \tilde{\nu} \rangle) \mid (l = \tilde{l} \wedge \nu = \tilde{\nu} \wedge l \in L \wedge \tilde{l} \in L) \vee (l = DupZoneMap(\tilde{l}) \wedge \nu = \tilde{\nu} \wedge \tilde{\nu} \in inv(\tilde{l}) \wedge l \in L \wedge \tilde{l} \in \tilde{L})\}$

$\mathcal{M}$ 上の任意の l に対し，*1* 回の離散遷移で l に遷移可能な任意のロケーションを l_{prev} とする．また，任意のロケーション l において，分布中の遷移確率 p の遷移に対するガード式を $g_l(p)$ とする．$\langle l, \nu \rangle$ に対応する $\langle \tilde{l}, \nu \rangle$ は，R より，

(i) $(l = \tilde{l} \wedge \nu = \tilde{\nu} \wedge l \in L \wedge \tilde{l} \in L)$

(ii) $(l = DupZoneMap(\tilde{l}) \wedge \nu = \tilde{\nu} \wedge \tilde{\nu} \in inv(\tilde{l}) \wedge l \in L \wedge \tilde{l} \in \tilde{L})$

の *2* つのロケーションが存在する．以下では，$\langle l, \nu \rangle$ に対応する $\langle \tilde{l}, \tilde{\nu} \rangle$ を上記の番号によって表記する．

先ず，$\mathcal{M}$ 上の任意の遷移が，$\tilde{\mathcal{M}}$ に存在することを示す．

Case:$\langle l_{prev}, \nu \rangle$ における振る舞い

時間遷移に関しては自明．離散遷移は，$\langle l, \nu \rangle$ からの遷移は，アルゴリズムより $\tilde{\mathcal{M}}$ 上に必ず保持され，ガードや遷移する確率も同じ．よって，すべての振る舞いが $\mathcal{M}$ 上に存在する．

Case:$\langle l, \nu \rangle$ における振る舞い

(i) の場合．時間遷移に関しては自明．離散遷移は，$\langle l, \nu \rangle$ と同様に，すべての振る舞いが $\mathcal{M}$ 上に存在する．*(ii)* の場合．時間遷移について考える．$\langle l, \nu \rangle$ の時間遷移は $\nu + t \in inv(l)$ である t において可能である．これに対し，$inv(\tilde{l})$ の定義より，$\nu \in inv(\tilde{l})$ が存在する場合，$\tilde{l}$ への任意の遷移は元モデル $\mathcal{M}$ に存在する．

Case:$\langle \tilde{l}, \nu \rangle$ における振る舞い

時間遷移について．

$\langle \tilde{l}, \nu \rangle$ で時間遷移可能なゾーンは，上述の $inv(\tilde{l})$ によって示される．複製元のロケーション l に対し，明らかに $inv(\tilde{l}) \in inv(l)$ という関係が成り立つ．よって，$\tilde{l}$ 上の任意の時間遷移が，$\mathcal{M}$ 上の l で成立する．$\nu + t \in inv(\tilde{l})$ となる同様の時間遷移が存在する．離散遷移は，アルゴリズムより，$\tilde{\mathcal{M}}$ では $inv(\tilde{l})$ を満たす l 上の遷移を $\tilde{l}$ に保存するため，$inv(\tilde{l})$ を満たす l 上の遷移は，$\tilde{\mathcal{M}}$ 上に必ず存在する．

続いて，$\tilde{\mathcal{M}}$ 上の任意の遷移が，$\mathcal{M}$ に存在することを示す．

Case:$\langle \tilde{l}_{prev}, \nu \rangle$ における振る舞い

時間遷移について．

アルゴリズムより，$\tilde{l}_{prev}$ と対応するロケーション l_{prev} が元モデル上に存在する．

(i) の $\tilde{l}$ に関しては自明なため，以下では $\tilde{l}$ は *(ii)* であるとする．

離散遷移について．

$\tilde{l}_{prev}$ から $\tilde{l}$ への離散遷移が実行可能なゾーン $\zeta^{\tilde{l}}_{enab}$ は，$well\text{-}formed$ より，$(inv(\tilde{l}_{prev}) \wedge g_{\tilde{l}_{prev}}(p))[X := 0] \wedge inv(\tilde{l})$ を満たすものに限られる．l_{prev} から l へ離散遷移可能なゾーン ζ^{l}_{enab} は，$(inv(l_{prev}) \wedge g_{l_{prev}})[X := 0] \wedge inv(l)$ であり，定義より，明らかに $\zeta^{\tilde{l}}_{enab} \in \zeta^{l}_{enab}$ である．

離散遷移について．

アルゴリズムより，l 上で $inv(\tilde{l})$ を満たす $g_l(p)$ を持つ離散遷移が $\tilde{l}$ で再現されるため，同様の離散遷移が l で存在する．□

5 実験

5.1 実験の目的

本研究において提案する手法がどの程度スケーラビリティの低さを改善できているかを評価することが，この実験の目的となる．メモリ消費量の節約ができていれば，提案手法が有用であるということができる．以下では，実験環境と実験方法について述べた後，実験の対象となるモデル 2 つを紹介し，最後に実験結果として得られたデータを掲載する．

ただし，本研究における「複製ゾーンロケーションの生成」部に関しては未実装であるため，同時実行可能性が偽であるようなモデルには適用できない．以降に示す実験結果は，同時実行可能性はすべて真であったものである．

5.2 実験環境

今回の実験において利用した環境，ツールを以下に示す．

OS	:	CentOS 5.3
CPU	:	Pentium4 3.20GHz
Memory	:	2GB
PRISM	:	$version$ 3.3
Eclipse	:	$version$ 3.5.1
JDK	:	$version$ 1.6.0_b09

5.3 実験対象となる確率時間オートマトン

本研究で実験の対象としたモデルは，FireWire Root Contention Protocol[15] である．以下では，そのモデルの振る舞いや検査対象とする性質について簡潔に述べる．

IEEE1394 ハイパフォーマンスシリアルバスは，マルチメディアシステムやデバイスのネットワークとして，テレビやパソコンのようなデジタル機器で多く用いられている．このシステムは，多くの異なるプロトコルを，各用途に利用しているが，FireWire Root Contention Protocol（図 9）とは，含まれるプロトコルの一種である．

このプロトコルは，バスのリセットが行われた後で使用される．例えば，ネットワーク内のノード（デバイス）に追加や削除が行われるような場合である．バスをリセットし，その後にすべてのノードを同じ状態にする．このプロトコルの目的は，ネットワークトポロジが木構造になっているかどうかを検査することである．スパニング木を構成するために，リーダーがプロトコルによって選出される．

リーダーを選出するために，ノードは,「親になって下さい」という要求を隣同士で交換する．しかしながら，2 つのノードが同時に「親になって下さい」と送信した場合,「争い」が起きるかもしれない．この争いを打開する方法は，root contention と呼ばれる，確率的，時間的な要素を用いた次のような方法で解決する．各ノードがコインを振ることで，要求のために長い時間待機するか，もしくは短い時間待機するか，を選択する．このプロトコルの興味深い性質は，リーダーが deadline までに選出されるかどうか，である．

図 9 において示した確率時間オートマトンは，文献 [30] のモデルの確率的な拡張であり，簡単化されたモデルである．タイミングの制約は IEEE1394a の規格に従う．例えば，もしノードがロングワイヤーを使用しているならば，360ns の通信遅延が発生し，他のタイプのワイヤーをが異なる通信遅延を持つ場合，このパラメータを変化させることで検証を行うことができる．本研究において検証を行う性質は,「deadline までにリーダーが選出されない確率が p 未満である」とい内容で検証を行う．

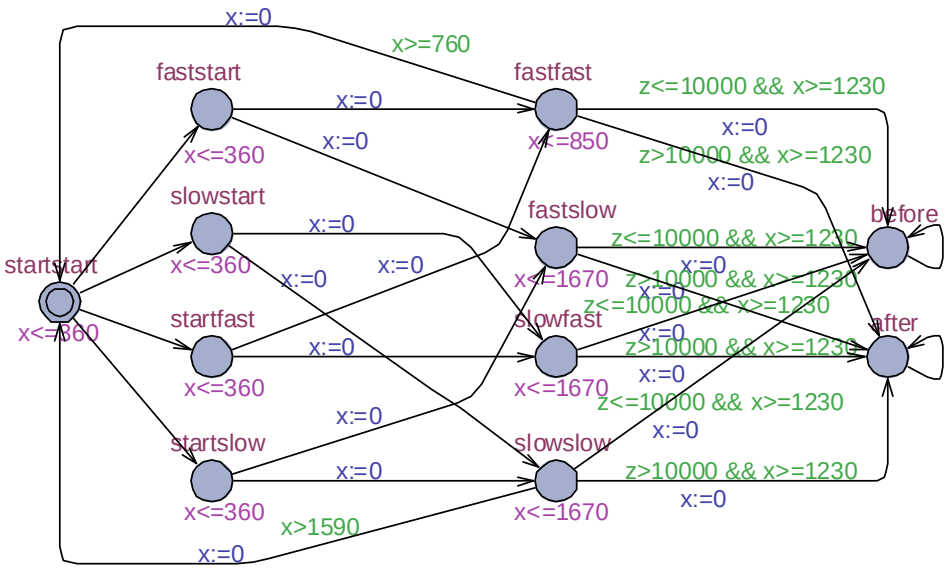

図 9: FireWire Root Contention Protocol (Abstract) Deadline100

5.4 実験方法

各モデルに対し，性質中の Deadline によってモデルサイズに大きく影響するため，Deadline を増やしていった場合に，モデル検査の状態数，モデルをビルドする時間，検査を実行する時間の観点で Digital clocks[19] との比較を行う．入力となる性質に含まれる閾値 p は，Digital clocks によってあらかじめ求められた最大確率の半分に設定し，入力とする．提案手法によ

表 1: FireWire Abst モデルの実験結果

D	Digital clocks[19]				proposal approach					Max reachability probability
	States	Time(s)		Mem (MB)	Loop	States	Time(s)		Mem (MB)	
		Build	MC				Build	MC		
50	298165	7.5	8.43	9.1	27	44	0.86	0.79	4.93	0.21875
100	686165	26.5	56.3	24.4	142	165	0.91	12.74	7.01	0.02526855
200	1462165	97.4	376.3	54.9	563	602	1.86	2233	10.03	0.00037044

D：DeadLine，States：モデルの状態数，Build：モデルのエンコード時間，
MC：モデル検査にかかった時間，Loop：CEGAR ループ回数，
Max reachability probability：最大到達確率をそれぞれ表す.

る実験では，k-最短路探索における k の値を $k = 10$，$k = 100$，$k = 1000$ の 3 通りとして行った.

提案手法における実行時間の計測は，java.lang.System クラスの currentTimeMills() を利用し，モデルのビルド時間，検査時間の両方を計測する．一方，メモリ消費量は，jconsole によるプロセスの監視によってデータを取得した．PRISM では，ログの出力にメモリ消費量や実行時間が出力されるため，その値を結果としてそのまま用いる．実験は 5 回ずつ行い，その平均値を結果として掲載する.

5.5 実験結果

実験結果を表 1，2 に示す．表 1 では，状態数，実行時間，メモリ消費における Digital clocks との比較を示している．なお，表 1 の提案手法の結果はパスの最大探索数 k を 1000 としている．D=50，100 のときは，状態数，実行時間，メモリ消費において提案手法が優れているという結果になった．一方で，D=200 のときは，パス数が 1000 の場合でも指定した確率値に達しなかったため，正しい結果が得られずに終了した．表 2 では，k の値を変えたときの提案手法の結果を示している．D=50，100 の場合は 1000 程度で十分であると考えられるが，D=200 の場合は最大値の半分も達していないため，反例を構成するパス数が非常に多くなると予想できる.

表 2: FireWire Abst モデルの実験結果（k 個の探索まで実行した場合）

D	k	proposal approach						Max reachability probability
		Loop	States	Time(s)		Mem(MB)	probability	
				Build	MC			
50	10	36	53	0.95	1.14	4.94	0.1445312	0.21875
	100	62	95	0.85	3.87	4.94	0.2187318	
	1000	62	95	1.00	39.30	6.40	0.2187499	
100	10	123	144	0.87	8.12	6.29	0.0046386	0.02526855
	100	157	185	1.13	18.57	7.00	0.0170898	
	1000	216	283	1.20	340.23	7.79	0.0252670	
200	10	473	504	1.84	500.70	9.43	0.0000045	0.00037044
	100	519	552	1.33	661.60	9.67	0.0000259	
	1000	563	602	1.87	2233.11	10.04	0.0000917	

k：k-最短路アルゴリズムによって探索するパスの数,
probability：k 個のパスによって計算された確率を表す

5.6 考察

5.6.1 メモリ消費量・状態数

表 1 を見ると，Digital clocks に比べて，最大で 5 分の 1 程度と大きくメモリ消費量が削減されていることがわかる．提案手法では，確率時間オートマトン中のクロック変数に関する制約をすべて除去したモデルであり，クロック変数制約をそのまま自然数として表したモデルよりも状態数が非常小さくなっているためであると考えられる．しかし，メモリ消費量は状態数程の差は見られない．これは，k-最短路によって複数個のパスを探すためのメモリがある程度必要であることが根拠として挙げられる．

5.6.2 実行時間

モデル検査時間に関しても，Digital clocks に比べ早くなっている．これは，状態数の大きな差によるものであると考えられる．一方で，D=200 でパスの数が 10 の場合でも計算時間が多く掛っているが，Loop 回数や状態数も初期抽象化の段階より 10 倍以上に増加しており，これは最短パス近辺で洗練が実行され続けている，ということが言える．本実験において入力した性質は，リーダーを選出するまでに deadline を超えてしまう，という内容であるため，反例として探索するパスは，deadline を超えるパスである．D=200 では，dealine を超

えるパスは，目的ロケーションに到達するまでにモデルのある地点を何度かループする必要があり，最短路はループを行うような長いパスではないため，その点で洗練が多く行われていると考えられる．

5.6.3 パス数

deadline が 50，100 の場合は 1000 程度で十分であると考えられるが，200 の場合は最大値の半分も達していない．ただし，deadline が 200 の場合，探索パス数の上限値まで正しい反例を求めても，入力された p 値に届かず，検査をきちんと実行できなかった．これは，deadline の値を増加させることで，実行時間において述べたように，各パスは deadline を超えてしまうような経路を選択する必要がある．よって，それぞれのパスの重みは小さくなり，有効な反例とするに必要なパス数が増えているためと考えられる．

6 評価

評価では，実験結果を踏まえた今後の課題について言及する．

実験結果では，全体的に状態数の削減が目立ち，状態数削減では有効な手法であると考えられる．ただし，反例に必要なパス数が増加すると，計算量が増大し，実行時間が増えてしまった．これは，パス数によって k-最短路のコストが指数的に増えているためであると考えられる．

実験結果の D=200 の場合のように，入力された性質の反例において必要なパス数が多い場合が存在する．提案手法では k 個のパスが必要な反例を探索するため，この場合は反例となるパスを探索しない従来手法に比べて，どのようにしても効率化を図ることが難しい．

また，実際には $false$ であるにも関わらず，k 個のパスで反例が入力の p（p 値とする）を超えなかった場合に $true$ と出力してしまう，信頼性の問題が浮上する．本手法では，抽象モデル上の確率的モデル検査を ValueIteration を用いて行っている．この ValueIteration ではアドバーサリを求めるだけでなく，抽象モデル上の到達確率（v 値とする）を出力する．よって，v 値が p 値よりも小さい場合，当然 $true$ として終了だが，v 値が p 値よりも超えている場合,「正しい答えが得られなかった」と警告を出力して終了することで，検査結果の信頼性を保持することができる．

さらに，反例を構成するパスが多くなる場合を判定し，ハイブリッドに検査を実行する手法が有用ではないかと考えられる．パス数が増大する場合，反例を出力はできないが，従来の ValueIteration のような数値解析の手法によって検査結果を判定し，そうでない場合に提案手法を用いて反例解析を行う，という手法である．

ここで，反例を構成するパスが多くなる場合をどのように判定するかであるが，単純には，以下の 2 つの方法が考えられる．

- k の値に上限を定める
- 最短パスの重みによって判断

最短パスの重みによって判断する方法とは，具体的には，性質として入力された p 値に対し，最初に探索されたパスの確率の値が非常に小さい場合などで判断する．現在は k の上限を定めることによってアルゴリズムが終了する．今後は，最短パスの重みの評価することでハイブリッド手法を展開していくことを考えたい．

7 あとがき

本研究では，時間オートマトンの時間抽象化とその洗練手法を拡張し，確率時間オートマトンの CEGAR ループに基づくフレームワークを提案した．また，反例となるパス群の同時実行可能性について言及し，同時に起こり得ないパスが抽象モデル上で発生しないようにするために，具体モデルの等価変換を行うことで解決した．加えて，完全な実装ではないが，同時実行可能性以外の面を実装したツールを用いて，確率時間オートマトンの CEGAR ループに対する評価実験を行った．

評価実験では，実用的な例題 FireWire Root Contention Protocol を用いて，Digital clocks との比較を行った．パスの数が多くない場合だと，実行時間・メモリ消費量の点で優れていることがわかった．不具合の箇所を特定するのに有用な反例を導出することができる．よって，6 で述べたようなハイブリッドな検査手法が有効であると考えられる．

今後の課題としては，提案手法の完全な実装が挙げられる．提案手法では，出発条件のパターンを区別するためにゾーンに対する `not` 演算が必要である．一般的な DBM では `not` 演算に対応していないため，現段階では完全な実装を行うことができない．`not` 演算に関する既存研究を用いるなどして，解決していきたい．

謝辞

本研究を行うにあたり，理解あるご指導を賜り，常に励まして頂きました楠本真二教授に心から感謝申し上げます.

本研究の全過程を通じ，適切かつ丁寧なご指導を頂きました岡野浩三准教授に深く感謝申し上げます.

本研究において，常に適切なご助言およびご指導を頂きました肥後芳樹助教に深く感謝致します.

本研究に多大なるご助言およびご指導を頂きました柿元健特任助教に深く感謝致します.

本研究の全過程を通じ，多大な協力，助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻博士後期課程 2 年長岡武志氏に深く感謝申し上げます.

本研究について熱心に協力を頂いた大阪大学大学院情報科学研究科コンピュータサイエンス専攻博士前期課程 1 年田中俊彰氏に感謝します.

その他楠本研究室の皆様のご協力に，心より感謝致します.

最後に，コンピュータサイエンス専攻にてお世話頂きました諸先生方にお礼申し上げます.

参考文献

[1] H. Aljazzar and S. Leue. Counterexamples for model checking of markov decision processes. *Technical Report soft-08-01, Chair for Software Engineering*, 2007.

[2] R. Alur and D.L. Dill. A theory of timed automata. *In Theoretical Computer Science*, Vol. 125, pp. 183–235, 1994.

[3] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time markov chains. *In R. Alur and T. Henzinger, editers, Proc. 8th International Conference on Computer Aided Verification (CAV'95)*, Vol. 1102 of LNCS, pp. 269–276, 1996.

[4] C. Baier and J.P. Katoen. Principles of model checking (representation and mind series). *The MIT Press*, 2008.

[5] D.P. Bertsekas. Dynamic programming and optimal control. *Athena Scientific*, 1995.

[6] E.M. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logics. *ACM Transactions on Programming Languages and Systems*, Vol. 8(2), pp. 244–263, 1986.

[7] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. *International Conference on Computer Aided Verification (CAV'00)*, pp. 154–169, 2000.

[8] E.M. Clarke, O. Grumberg, and D.A. Peled. Model checking. *The MIT Press*, 1999.

[9] A. David, J. Hakansson, K. G. Larsen, and P. Petterson. Model checking timed automata with priorities using dbm subtraction. *Lecture Notes in Computer Science*, Vol. 2402, pp. 128–142, 2006.

[10] C. Daws, M. Kwiatkowska, and G. Norman. Automatic verification of the ieee 1394 root contention protocol with kronos and prism. *International Journal on Software Tools for Technology Transfer (STTT)*, Vol. 5(2), pp. 221–236, 2004.

[11] C. Dawsm, A. Olivero, S. Tripakis, and S. Yovine. The tool kronos. *In R. Alur, T. A. Henzinger and E. D. Sontag, editers, Hybrid Systems III, Lecture Notes in Computer Science*, Vol. 1066, pp. 208–219, 1336.

[12] M. Fruth. Probabilistic model checking of contention resolution in the ieee 802.15.4 low-rate wireless personal area network protocol. *In Proc. 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISOLA'06)*, November 2006.

[13] T. Han and J. Katoen. Counterexamples in probabilistic model checking. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 72–86, 2007.

[14] H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, Vol. 6, pp. 512–535, 1994.

[15] M. Kwiatkowska, G. Norman, and J.Sproston. Probabilistic model checking of deadline properties in the ieee1394 firewire root contention protocol. *Formal Aspects of Computing*, Vol. 14(3), pp. 295–318, 2003.

[16] M. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. *In Proc. 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pp. 322–323, 2004.

[17] M. Kwiatkowska, G. Norman, and D. Parker. Game-based abstraction for markov decision processes. *In Proc. 3rd International Conference on Quantitative Evaluation of Systems (QEST'06), pages 157-166, IEEE CS Press. Winner of the QEST'06 Best Paper Award.*, 2006.

[18] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic games for verification of probabilistic timed automata. *In Proc. 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'09), LNCS*, Vol. 5813, pp. 212–227, Springer, 2009.

[19] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, Vol. 29, pp. 33–78, 2006.

[20] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretial Computer Science*, Vol. 282, pp. 101–150, 2002.

[21] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the ieee 802.11 wireless local area network protocol. *In H. Hermanns and R. Segala (editors) Proc.*

PAPM/PROBMIV'02, volume 2399 of Lecture Notes in Computer Science, Vol. 2399, pp. 169–187, 2002.

[22] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Inf. and Comp.*, Vol. 205(7), pp. 1027–1077, 2007.

[23] T. Nagaoka. Model abstraction of timed automata based on counterexample-guided abstraction refinement loop. *Department of Computer Science, Graduate School of Information Scinece and Technology*, 2008.

[24] T. Nagaoka, A. Ito, K. Okano, and S. Kusumoto. Qos evaluation for real-time distributed systems using the probabilistic model checker prism. *In Proceedings of International Workshop on Informatics*, pp. 60–66, 2009.

[25] T. Nagaoka, K. Okano, and S. Kusumoto. Abstraction of timed automata based on counterexample-guided abstraction refinement loop. *IEICE Technical Report*, Vol. 107, No. 505, pp. 103–108, 2008.

[26] T. Nagaoka, K. Okano, and S. Kusumoto. An abstraction refinement technique for timed automata based on counterexample-guided abstraction refinement loop. *IEICE Transactions on Information and Systems*, Vol. E93-D, No. 5, p. to appear, 2010.

[27] D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.

[28] M Puter. Markov decision processes: Discrete stochastic dynamic programming. *Wiley Interscience*, 1994.

[29] P. Roy, D. Parker, G. Norman, and L. de Alfaro. Symbolic magnifying lens abstraction in markov decision processes. *In Proc. 5th International Conference on Quantitative Evaluation of Systems (QEST'08), pages 103-112, IEEE CS Press*, 2008.

[30] D. Simons and M. Stoelinga. Mechanical verification of the ieee 1394a root contention protocol using uppaal2k. *Springer International Journal of Software Tools for Technology Transger*, 2001.